Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Λειτουργικά Συστήματα (Κ22) / Περίοδος 2025-2026
2η Εργασία

Υποστήριξη MLFQ scheduling xv6

Ο scheduler στο xv6 είναι απλός round robin που εκτελεί τις διεργασίες κυκλικά. Σε αυτή την εργασία, θα υλοποιηθεί ένας scheduler που υποστηρίζει το μοντέλο MLFQ (Multilevel Feedback Queue).

Εισαγωγή νέων κλήσεων συστήματος

Για την εισαγωγή νέων κλήσεων συστήματος στο xv6, θα χρειαστεί να κατανοήσετε και να τροποποιήσετε τα παρακάτω αρχεία::

proc.c, proc.h, syscall.c, syscall.h, sysproc.c, user.h, and usys.pl.

Συγκεκριμένα:

- `user.h` - περιέχει τις δηλώσεις των κλήσεων συστήματος..
- `usys.pl` - παράγει μία λίστα με το σύνολο των κλήσεων συστήματος που εξάγει ο πυρήνας.
- `syscall.h` - περιέχει την αντιστοίχιση ονόματος κλήσης συστήματος σε αριθμό κλήσης συστήματος. Πρέπει να προστεθούν νέοι αριθμοί για νέες κλήσεις συστήματος
- `syscall.c` - περιέχει βοηθητικές συναρτήσεις για τα ορίσματα των κλήσεων συστήματος και δείκτες προς τις υλοποιήσεις τους
- `sysproc.c` - περιέχει υλοποιήσεις των κλήσεων συστήματος που σχετίζονται με διεργασίες.
- `proc.h` - περιέχει τη δομή `struct proc`. Θα χρειαστεί να επεκταθεί ώστε να αποθηκεύει επιπλέον πληροφορίες για κάθε διεργασίες που θα είναι απαραίτητες
- `proc.c` - περιέχει τη συνάρτηση `scheduler()` που υλοποιεί τον χρονοπρογραμματισμό και την αλλαγή μεταξύ διεργασιών

Θα υλοποιήσετε την εξής νέα κλήση συστήματος στο xv6

1. Κλήση συστήματος `int getpinfo(struct pstat *)`. Μέσω αυτής της κλήσης συστήματος θα αντιγράφει στο `struct pstat *` πληροφορίες για κάθε ενεργή διεργασία (pid, ppid, όνομα, προτεραιότητα, κατάσταση, μέγεθος και ό,τι άλλο μπορεί να χρειαστεί). Η δομή αυτή θα οριστεί σύμφωνα με την `struct proc`.

Για επιτυχή εκτέλεση επιστρέφουν οι κλήσεις συστήματος επιστρέφουν 0, αλλιώς επιστρέφουν -1.

Για την υποστήριξη προτεραιότητας, θα πρέπει να επεκταθεί η `struct proc` ώστε να υποστηρίζει προτεραιότητες.

Υλοποίηση προγράμματος χρήστη ps

Το πρόγραμμα σας θα υλοποιεί παρόμοια λειτουργία με την εντολή `ps` στο Linux, δηλαδή θα δείχνει πληροφορίες σχετικά με τις ενεργές διεργασίες στο xv6. Για το λόγο αυτό θα υλοποιήσετε το `ps.c` πρόγραμμα στον κατάλογο `user`, που να χρησιμοποιεί την κλήση συστήματος `getpinfo(struct pstat *)`. Στη συνέχεια θα χρησιμοποιείται από το πρόγραμμα η δομή `struct pstat` που έχει γεμίσει από την κλήση συστήματος. Οι πληροφορίες για κάθε διεργασία θα περιλαμβάνουν και το επίπεδο προτεραιότητας κάθε διεργασίας.

Υλοποίηση MLFQ scheduler

Η βασική ιδέα είναι απλή. Κατασκευάστε έναν απλοποιημένο MLFQ scheduler με 4 ουρές προτεραιότητας ως εξής. Η ουρά με αριθμό 0 έχει την υψηλότερη προτεραιότητα και η ουρά με αριθμό 3 έχει τη χαμηλότερη προτεραιότητα. Όταν μία διεργασία χρησιμοποιήσει ολόκληρο το μερίδιο της (που μετράται ως αριθμός από ticks), θα πρέπει να πάει στην ουρά της αμέσως επόμενης (χαμηλότερης) προτεραιότητας. Μέσα σε κάθε ουρά, η μέθοδος χρονοπρογραμματισμού΄θα πρέπει να είναι round-robin.

Για να δοκιμάσετε αρχικά την υλοποίηση σας, εκτελέστε το xv6 αρχικά σε μία μόνο CPU. Μπορείτε να το κάνετε εκτελώντας:

```
$ CPUS=1 make qemu
```

Υλοποίηση MLFQ

Ο MLFQ scheduler θα πρέπει να ακολουθεί τους εξής κανόνες:

1. 4 επίπεδα προτεραιότητας, αριθμημένα από το 0 (υψηλότερη) ως το 3 (χαμηλότερη)
2. Όποτε συμβαίνει το tick των 10 ms από τον μετρητή του xv6, η διεργασία υψηλότερης προτεραιότητας θα πρέπει να εκτελεστεί.
3. Η διεργασία υψηλότερης προτεραιότητας που είναι έτοιμη θα χρονοπρογραμματιστεί να εκτελεστεί όποτε η προηγουμένως εκτελούμενη διεργασία τερματίζει, μπλοκάρεται, κοιμάται ή με κάποιον τρόπο παραδίδει τη CPU.
4. Αν υπάρχουν πάνω από μία διεργασίες σε ένα επίπεδο προτεραιότητας, τότε ο scheduler θα πρέπει να χρονοπρογραμματίσει όλες τις διεργασίες στο επίπεδο αυτό με τρόπο round robin.
5. Όταν συμβαίνει ένα timer tick, όποια διεργασία χρησιμοποιούσε τη CPU, θα πρέπει να θεωρείται ότι χρησιμοποίησε ένα ολόκληρο timer tick της CPU, ακόμη και αν δεν ξεκίνησε στο προηγούμενο tick. (Σημείωση: το timer tick είναι διαφορετικό από το χρονομερίδιο της διεργασίας).
6. Το χρονομερίδιο των διεργασιών με προτεραιότητα 0 είναι 4 timer ticks, με προτεραιότητα 1 είναι 8 είναι timer ticks, με προτεραιότητα 2 είναι 16 timer ticks και με προτεραιότητα 3 είναι 32 timer ticks.
7. Όταν ξεκινάει νέα διεργασία, θα εντάσσεται στο επίπεδο 0.

8. Αν δεν ξεκινήσει διεργασία υψηλότερης προτεραιότητας και η εκτελούμενη διεργασία δεν παραδώσει τη CPU, τότε η διεργασία αυτή χρονοπρογραμματίζεται για ένα ολόκληρο χρονομερίδιο πριν ο scheduler δώσει τη CPU σε άλλη διεργασία.
9. Στις προτεραιότητες 0, 1 και 2, αφού μία διεργασία καταναλώσει το χρονομερίδιο της, θα πρέπει να πάει σε ουρά χαμηλότερης προτεραιότητας.
10. Αν μία διεργασία εθελοντικά παραδώσει τη CPU πριν εκπνεύσει το χρονομερίδιο της σε ένα συγκεκριμένο επίπεδο προτεραιότητας, το χρονομερίδιο δεν πρέπει να επανέλθει στην αρχική τιμή. Την επόμενη φορά που θα χρονοπρογραμματιστεί η διεργασία, θα συνεχίσει να χρησιμοποιεί το υπόλοιπο του χρονομεριδίου στο συγκεκριμένο επίπεδο προτεραιότητας.
11. Για να αποφευχθεί το πρόβλημα της λιμοκτονίας (starvation), θα υλοποιηθεί ένας μηχανισμός αύξησης προτεραιότητας. Αν μία διεργασία περιμένει 10 φορές το χρονομερίδιο της στο τρέχον επίπεδο προτεραιότητας, θα αυξηθεί το επίπεδο προτεραιότητας στο επόμενο (εκτός αν είναι ήδη στο επίπεδο 0.

Όταν ολοκληρώσετε την εργασία, ο πυρήνας θα πρέπει να περνάει όλα τα τεστ στα προγράμματα `usertests`, δηλαδή:

```
$ usertests
...
ALL TESTS PASSED
$
```

Κώδικας

Χρησιμοποιήσετε τον κώδικα για την εργασία από το βασικό αποθετήριο του xv6 https://github.com/mit-pdos/xv6-riscv.

Τα αρχεία που θα χρειαστείτε για αυτή την εργασία διανέμονται μέσω του συστήματος ελέγχου πηγαίου κώδικα git. Μπορείτε να βρείτε πληροφορίες για το git στο βιβλίο git ή σε άλλες δημόσιες πηγές. Το git επιτρέπει να διατηρείτε πληροφορία για όλες τις αλλαγές που έχετε κάνει στον κώδικα. Για παράδειγμα, αν τελειώσετε ένα μέρος της εργασίας και θέλετε να καταχωρήσετε τοπικά τις αλλαγές σας, μπορείτε να καταγράψετε (commit) τις αλλαγές σας μέσω της εντολής

```
$ git commit -am 'my solution for k22 project'
Created commit 60d2135: my solution for k22 project
 1 files changed, 1 insertions(+), 0 deletions(-)
$
```

Ημερομηνία Παράδοσης: 18 Ιαν 2026
Τρόπος παράδοσης: υποβολή στο eclass, θα πρέπει να παραδοθεί ένα αρχείο tar με περιεχόμενο όλα τα σχετικά αρχεία.
Συνοδευτικό υλικό: τεκμηρίωση που να εξηγεί τον τρόπο με τον οποίο εργαστήκατε.
Υλοποίηση: η εργασία είναι ατομική.
Η εργασία θα εξεταστεί σε συμβατό xv6 emulator (Linux, Windows WSL) με πρόγραμμα που θα ανακοινωθεί μετά την ημερομηνία παράδοσης.